Leap+Verify: Regime-Adaptive Speculative Weight Prediction for Accelerating Neural Network Training

Jeremy McEntire

Abstract

We introduce **Leap+Verify**, a framework that applies speculative execution—predicting future model weights and validating predictions before acceptance—to accelerate neural network training. Inspired by speculative decoding in language model inference and by the Automatically Scalable Computation (ASC) architecture for program execution, Leap+Verify decomposes training into three dynamically detected regimes (chaotic, transition, stable) using activation-space cosine similarity as a real-time Lyapunov proxy signal. Within each regime, analytic weight predictors (momentum, linear, quadratic extrapolation) attempt to forecast model parameters K training steps ahead; predictions are accepted only when validated against a held-out loss criterion.

We evaluate Leap+Verify on GPT-2 124M, Qwen 2.5-1.5B, and Qwen 2.5-7B trained on WikiText-103 across five random seeds each, sweeping prediction depth $K \in \{5, 10, 25, 50, 75, 100\}$. Momentum-based prediction (Adam moment extrapolation) fails catastrophically at *all three* scales, with predicted losses exceeding actuals by $100-10{,}000\times$—a universal **norm explosion** in optimizer-state extrapolation. Finite-difference predictors (linear, quadratic) succeed where momentum fails: at 124M, they achieve 24% strict acceptance at K=5 in stable regimes; at 1.5B, they achieve 37% strict acceptance in transition regimes; at 7B, linear prediction achieves 60–90% strict acceptance across all K values in stable regime. The **scale-dependent** findings are twofold. First, regime distribution shifts dramatically: GPT-2 124M spends 34% of training in stable regime, Qwen 1.5B spends 64% in chaotic regime, while Qwen 7B reverses the 1.5B trend—spending 48% of training in chaotic but reaching stable regime by step 1350 in all seeds. Second, at 7B the momentum catastrophe *tightens*: inflation concentrates in a narrow $143-327\times$ band at K=5 (vs. $122-173\times$ at smaller scales), while linear prediction achieves 60–90% strict acceptance—its highest rates. Cross-seed results are highly consistent (<2.5% validation loss CV at all scales), and the three-regime framework produces identical phase boundaries (±50 steps) across seeds.

1 Introduction

Training modern neural networks is an inherently sequential process: each gradient update depends on the current parameter state, which depends on all prior updates. This sequential bottleneck consumes enormous computational resources—training frontier language models requires thousands of GPU-hours and millions of dollars [Kaplan et al., 2020]. Yet the sequential nature of gradient descent is not as rigid as it appears. Weight trajectories exhibit substantial regularity, particularly in late training, suggesting that future parameter states may be predictable from past trajectory information.

Speculative execution offers a principled way to exploit such predictability. In speculative decoding for language model inference [Leviathan et al., 2023, Chen et al., 2023], a small draft model generates candidate tokens that the target model verifies in parallel, achieving 2–3× speedup without altering the output distribution. The key insight is the *verify-then-accept* mechanism: predictions are cheap to generate and cheap to validate, so even moderate prediction accuracy yields substantial speedup.

We transplant this mechanism from inference to training. **Leap+Verify** predicts future model weights K steps ahead using analytic extrapolation of the weight trajectory, then validates the predicted weights against a held-out loss criterion before accepting the advance. The name derives from the Automatically Scalable Computation (ASC) architecture [Waterland et al., 2014], which accelerates sequential program execution by speculatively predicting future program states and caching verified state transitions. In ASC, a *recognizer* identifies predictable states, *predictors* forecast future states, and a *validator* confirms correctness before fast-forwarding execution. Leap+Verify instantiates each component for neural network training: the recognizer is an activation-space regime detector, the predictors are analytic weight extrapolators, and the validator is a held-out loss comparison.

A central finding is that prediction viability depends critically on the *training regime*. We identify three regimes—chaotic, transition, and stable—using cosine similarity between consecutive activation fingerprints as a proxy for the local Lyapunov exponent. Prediction is viable only in transition and stable regimes, where weight trajectories exhibit sufficient regularity. This regime dependence motivates *conditional* prediction: the system predicts only when the regime detector indicates favorable conditions, avoiding wasted computation during chaotic training phases.

Our most striking empirical finding is a **universal momentum catastrophe**: Adam moment extrapolation produces weight predictions 100−10,000× worse than actual validation loss at both 124M and 1.5B parameters. This catastrophe is caused by norm explosion—the extrapolated displacement $K \cdot m_t/\sqrt{v_t}$ far exceeds the region of validity around the current point on the loss surface. Finite-difference predictors (linear, quadratic), which extrapolate from actually observed checkpoint deltas, avoid this failure and achieve 9–37% strict acceptance at K=5 depending on regime and scale, with proximity-based acceptance reaching 100% at short horizons. The **scale-dependent** finding is that the distribution of training regimes shifts dramatically: at 1.5B, training remains chaotic for 64% of checkpoints (vs. 4% at 124M), severely limiting when prediction is viable. This regime-availability bottleneck has not been identified in prior work on weight nowcasting [Jang and Han, 2023, Knyazev et al., 2025, Guan et al., 2024].

Contributions.

1. **Leap+Verify**: A verify-then-accept mechanism for speculative weight prediction during training, inspired by speculative decoding and ASC.
2. **Regime-conditional prediction**: Three dynamically detected training regimes (chaotic, transition, stable) that determine when and how to predict.
3. **Universal momentum catastrophe and scale-dependent regime distribution**: Empirical demonstration that optimizer-state extrapolation fails at all scales, while the regime availability bottleneck varies non-monotonically with model size.
4. **Three-point scaling curve**: Results across GPT-2 124M, Qwen 2.5-1.5B, and Qwen 2.5-7B reveal that 7B models re-enter stable regime (unlike 1.5B), enabling high-acceptance linear prediction.
5. **Reproducible evaluation**: Five-seed experiments with consistent results at each of three model scales on WikiText-103.

2 Background and Motivation

2.1 Automatically Scalable Computation

The Automatically Scalable Computation (ASC) architecture [Waterland et al., 2014, 2013] accelerates sequential program execution by viewing it as a trajectory through state space. The architecture has three core components: (1) a *recognizer* that identifies states amenable to prediction, (2) a set of *predictors* that learn to forecast future states from observed trajectory structure, and (3) a *trajectory cache* that stores verified start-state/end-state pairs for fast-forwarding. When the recognizer identifies a predictable state, predictors generate candidate future states, speculative threads execute from those candidates, and results are cached. The sequential execution thread periodically queries the cache; on a match, it fast-forwards to the cached end state, achieving speedup proportional to prediction accuracy and trajectory length [Waterland et al., 2014].

ASC was originally implemented for x86 binary programs, where state vectors represent processor registers and memory. The present work adapts the architecture for neural network training, where the "state vector" is the model's parameter tensor and the "transition function" is a gradient update step. The exponentially large state space of parameters becomes tractable because training trajectories, like program trajectories, exhibit regular structure that predictors can exploit.

2.2 Speculative Decoding

Speculative decoding [Leviathan et al., 2023, Chen et al., 2023, Stern et al., 2018] accelerates autoregressive language model inference by using a small draft model to generate candidate token sequences that the target model verifies in parallel. The critical property is that verification is cheaper than generation: checking whether a sequence of K tokens is acceptable requires a single forward pass, while generating them sequentially requires K passes. Acceptance is governed by a rejection sampling scheme that preserves the target distribution exactly.

Leap+Verify adapts the predict-then-verify template to training. The "draft model" is an analytic weight extrapolator (momentum, linear, or quadratic); the "verification" is a loss evaluation on held-out data. Unlike speculative decoding, where acceptance is binary and distribution-preserving, Leap+Verify uses graded acceptance criteria (strict, adaptive, proximity-based) that trade off between conservatism and skip distance.

2.3 Training Dynamics and Phase Transitions

Neural network training traverses qualitatively different regimes. Cohen et al. [2021] identified two phases in full-batch gradient descent: progressive sharpening (where the maximum Hessian eigenvalue rises to $2/\eta$) and edge-of-stability (where sharpness oscillates at this threshold). Lewkowycz et al. [2020] documented a "catapult phase" where loss temporarily grows before decreasing. Frankle et al. [2020] showed that networks become "stable to SGD noise" early in training, after which models sharing a common training prefix converge to linearly connected solutions.

These observations suggest a multi-phase training structure, but prior work detects phases using expensive Hessian computations [Cohen et al., 2021] or post-hoc weight-space analysis [Frankle et al., 2020]. We use activation-space cosine similarity as an efficient, real-time proxy that requires only forward passes on a fixed probe set.

2.4 Weight Nowcasting

Several lines of work predict future weights during training. Kamarthi and Pittner [1999] used Taylor series extrapolation of weight trajectories. Sinha et al. [2017] trained a neural network to forecast future weights from weight history. Jang and Han [2023] designed the Weight Nowcaster Network (WNN), predicting 5 epochs ahead from 5 epochs of history. Knyazev et al. [2025] improved this with graph neural networks modeling neuron connectivity (NiNo), achieving up to 50% training acceleration. Guan et al. [2024] proposed XGrad, predicting future weights using optimizer update rules.

All existing weight prediction methods apply predictions *unconditionally*—they do not verify predictions before acceptance, and they do not condition on detected training regimes. Leap+Verify introduces both mechanisms: verify-then-accept prevents bad predictions from corrupting training, and regime conditioning restricts prediction to phases where trajectories are sufficiently regular.

3 Method

3.1 Regime Detection via Activation Fingerprinting

We detect training regimes by measuring the stability of the model's internal representations. At each checkpoint (every 50 training steps), we compute an *activation fingerprint*: the concatenated final hidden states produced by a fixed set of 100 probe sentences. The cosine similarity between consecutive fingerprints serves as a proxy for the local Lyapunov exponent of the training trajectory.

Formally, let $\mathbf{a}_t$ denote the activation fingerprint at step t. We compute:

$$s_t = \frac{\mathbf{a}_t \cdot \mathbf{a}_{t-\Delta}}{\|\mathbf{a}_t\| \|\mathbf{a}_{t-\Delta}\|} \tag{1}$$

where $\Delta = 50$ (the checkpoint interval). We classify regimes using thresholds derived from initial training runs:

$$\text{regime}(t) = \begin{cases} \text{stable} & \text{if } s_t > \tau_{\text{high}} \\ \text{chaotic} & \text{if } s_t < \tau_{\text{low}} \\ \text{transition} & \text{otherwise} \end{cases} \tag{2}$$

where τ_{high} and τ_{low} are averaged across initial seeds.

The cosine similarity signal offers several advantages over alternatives. Unlike Hessian eigenvalue computation [Cohen et al., 2021], it requires only forward passes (no second-order gradients). Unlike weight-space linear interpolation [Frankle et al., 2020], it operates in the representation space where functional similarity is more directly measured. The probe set is fixed across all checkpoints, ensuring that changes in s_t reflect changes in the model's representations rather than changes in input distribution.

3.2 Speculative Weight Prediction

Given a checkpoint at step t with parameters θ_t, we predict parameters θ_{t+K} using three analytic predictors that exploit different trajectory properties:

Momentum prediction. Uses the exponential moving averages maintained by the Adam optimizer:

$$\hat{\theta}_{t+K}^{\text{mom}} = \theta_t + K \cdot \frac{m_t}{\sqrt{v_t} + \epsilon} \tag{3}$$

where m_t and v_t are Adam's first and second moment estimates. This extrapolates the current update direction at constant velocity.

Linear prediction. Fits a line to two consecutive checkpoint states:

$$\hat{\theta}_{t+K}^{\text{lin}} = \theta_t + \frac{K}{\Delta}(\theta_t - \theta_{t-\Delta}) \tag{4}$$

This extrapolates the finite-difference velocity of the parameter trajectory.

Quadratic prediction. Fits a parabola through three consecutive checkpoints:

$$\hat{\theta}_{t+K}^{\text{quad}} = \theta_t + \frac{K}{\Delta}(\theta_t - \theta_{t-\Delta}) + \frac{K(K-\Delta)}{2\Delta^2}(\theta_t - 2\theta_{t-\Delta} + \theta_{t-2\Delta}) \tag{5}$$

This captures curvature in the trajectory, accounting for acceleration/deceleration.

After computing $\hat{\theta}_{t+K}$, we load the predicted weights into the model and evaluate the validation loss $\hat{L}_{t+K}$. We compare this against the current validation loss L_t using three acceptance criteria:

- **Strict**: Accept if $\hat{L}_{t+K} < L_t$ (predicted improvement).
- **Adaptive**: Accept if $\hat{L}_{t+K} < L_t + \sigma_L$ (within one standard deviation of recent validation losses).
- **Proximity (pct)**: Accept if $|\hat{L}_{t+K} - L_t| < \epsilon \cdot L_t$ (within a percentage of current loss).

If accepted, the model fast-forwards to step $t + K$, skipping K gradient updates. If rejected, training continues from step t with no modification—the prediction is purely speculative with no side effects.

3.3 Cascaded Prediction

To test deeper speculation, we chain multiple predictions in sequence. A cascade of depth D with step size K applies prediction D times, potentially advancing $D \times K$ steps. Each stage uses the predicted weights from the previous stage as its starting point. Cascades are evaluated only from stable-regime checkpoints, where individual prediction accuracy is highest.

4 Experimental Setup

4.1 Models and Data

We evaluate on three language models spanning two orders of magnitude:

- **GPT-2 124M**: 12 layers, 768 hidden, 12 heads. Randomly initialized, trained from scratch.
- **Qwen 2.5-1.5B** [Team, 2025]: 28 layers, 1536 hidden, 12 heads. Randomly initialized from pretrained architecture config, trained from scratch.
- **Qwen 2.5-7B** [Team, 2025]: 28 layers, 3584 hidden, 28 heads. Randomly initialized from pretrained architecture config, trained from scratch on A100 80GB.

All models are trained on WikiText-103 [Merity et al., 2017] with sequence length 256, using AdamW ($\eta = 5 \times 10^{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0.01) with cosine learning rate schedule and 100-step warmup, for 2000 steps.

4.2 Evaluation Protocol

For each model, we run five seeds (42–46) with identical hyperparameters. Each run proceeds in up to three passes:

1. **Pass 1 (Training)**: Train for 2000 steps, saving checkpoints every 50 steps (40 checkpoints per seed). Record activation fingerprints and regime classifications.
2. **Pass 2 (K-Sweep)**: For each non-chaotic checkpoint, evaluate all three predictors at $K \in \{5, 10, 25, 50, 75, 100\}$ with all three acceptance criteria. This yields up to $40 \times 3 \times 6 \times 3 = 2160$ evaluations per seed.
3. **Pass 3 (Cascades)**: From stable-regime checkpoints, evaluate cascaded predictions with configurations $(D, K) \in \{(4, 25), (2, 50), (10, 10)\}$. (Skipped at 7B due to memory constraints.)

5 Results

5.1 Training and Regime Detection

Table 1 summarizes the regime breakdown across all three model scales. All three models were trained for 2000 steps with checkpoints every 50 steps (40 checkpoints per seed). Regime classification is based on activation-space cosine similarity (Section 3.1).

Table 1: Regime breakdown by model scale (mean ± std across 5 seeds).

Model	**Chaotic**	**Transition**	**Stable**	**Unknown**
GPT-2 124M	1.6 ± 1.7	24.0 ± 2.7	13.4 ± 3.2	1.0 ± 0.0
Qwen 2.5-1.5B	25.6 ± 0.5	12.4 ± 1.1	1.0 ± 1.0	1.0 ± 0.0
Qwen 2.5-7B	19.2 ± 1.8	6.4 ± 1.1	13.4 ± 1.5	1.0 ± 0.0

The regime distribution shifts non-monotonically with model scale. GPT-2 124M spends 34% of training in stable regime and only 4% in chaotic. Qwen 2.5-1.5B reverses this: 64% chaotic, reaching stable in only 0–2 of 40 checkpoints. Qwen 2.5-7B presents a surprising reversal: despite being 4.7× larger than the 1.5B model, it spends 34% of training in stable regime—matching GPT-2 124M—while maintaining 48% chaotic. The chaotic-to-transition boundary is highly consistent across seeds at all scales (±50 steps), occurring at step ∼1000 for the 7B model. The first unknown checkpoint in each run reflects the initial state before consecutive fingerprints are available for similarity computation.

GPT-2 124M achieves a final validation loss of 1.616 ± 0.007 across seeds. Qwen 2.5-1.5B achieves 0.810 ± 0.019 (range: 0.797–0.843), with training times of 34.9 ± 0.7 minutes per seed on A100 40GB with mixed precision. Qwen 2.5-7B achieves 0.985 ± 0.013 (range: 0.970–1.002), with training times of ∼47 minutes per seed on A100 80GB.

5.2 K-Sweep: Speculative Depth vs. Acceptance Rate

Tables 2 and 3 present acceptance rates across prediction depth K, predictor type, and training regime. Evaluations are performed only on non-chaotic checkpoints (transition + stable).

Table 2: Strict acceptance rate (%) in transition regime: predicted loss must improve over current loss. Values are mean ± std across 5 seeds. N = total checkpoint evaluations across seeds.

K	Momentum	Linear	Quadratic
GPT-2 124M — Transition regime			
5	0.0 ± 0.0	9.3 ± 3.8	7.9 ± 3.0
10	0.0 ± 0.0	3.3 ± 3.2	3.4 ± 3.3
25	0.0 ± 0.0	0.7 ± 1.7	0.0 ± 0.0
GPT-2 124M — Stable regime			
5	10.0 ± 8.4	24.3 ± 6.8	22.1 ± 10.8
10	10.0 ± 8.4	18.8 ± 8.0	13.9 ± 8.4
25	10.0 ± 8.4	5.8 ± 3.5	9.1 ± 2.0
Qwen 2.5-1.5B — Transition regime			
5	0.0 ± 0.0	37.2 ± 11.0	36.5 ± 12.5
10	0.0 ± 0.0	31.3 ± 4.7	39.4 ± 9.1
25	0.0 ± 0.0	22.9 ± 8.1	20.5 ± 9.2
50	0.0 ± 0.0	17.8 ± 9.0	13.2 ± 6.3
75	0.0 ± 0.0	7.9 ± 8.4	6.7 ± 9.9
100	0.0 ± 0.0	6.2 ± 9.1	2.2 ± 5.0

Several patterns emerge across the 124M and 1.5B scales. First, momentum prediction achieves **0% strict acceptance in transition regimes** at both scales. In GPT-2's stable regime, momentum achieves ∼10% strict and ∼17% proximity acceptance—non-zero but still far below finite-difference predictors, and reflecting the small fraction of checkpoints where momentum displacement happens to land in a favorable region. Second, finite-difference predictors are substantially more effective in Qwen 1.5B's transition regime (37% strict at K=5) than in GPT-2's transition regime (9%)—larger models produce smoother trajectories in comparable regimes. Third, GPT-2's stable regime enables 22–24% strict acceptance at K=5, better than its own transition regime but below Qwen 1.5B's transition performance. Fourth, proximity-based acceptance reveals graceful degradation at both scales: ∼99–100% through K=5, declining with K. At 1.5B, proximity acceptance remains above 90% through K=25; at 124M, the decline is steeper (44% at K=25 in transition, 75% in stable).

The few Qwen 1.5B stable-regime checkpoints (N=5 total across seeds) show 100% proximity acceptance through K=25, consistent with GPT-2's stable-regime pattern.

5.2.1 7B Scale: Qwen 2.5-7B

The 7B model, unlike the 1.5B model, reliably reaches stable regime (mean 13.4 stable checkpoints per seed), enabling direct evaluation of all three predictors in the regime most favorable to prediction. Table 4 presents the K-sweep results.

Three findings distinguish the 7B results from the smaller scales.

Momentum catastrophe is universal but tightening. Momentum prediction achieves 0% acceptance at all K values across all 5 seeds, confirming the catastrophe scales to 7B. However, the inflation ratios at K=5 cluster in a narrower band (143−327×, mean ∼227×) than at 1.5B (173× mean) or 124M (122× mean). At K=100, 7B inflation reaches ∼2,700×—comparable to 1.5B but well below 124M's 10,764×. The tightening suggests that larger models' more uniform parameter distributions produce more predictable (though still catastrophic) momentum norm explosion.

Table 3: Proximity (pct) acceptance rate (%) in transition regime: predicted loss within 5% of current loss. Values are mean ± std across 5 seeds.

K	Momentum	Linear	Quadratic
GPT-2 124M — Transition regime			
5	0.0 ± 0.0	99.1 ± 1.9	100.0 ± 0.0
10	0.0 ± 0.0	97.3 ± 2.5	93.4 ± 4.2
25	0.0 ± 0.0	44.0 ± 9.8	37.8 ± 12.0
50	0.0 ± 0.0	14.1 ± 6.8	9.3 ± 7.1
GPT-2 124M — Stable regime			
5	16.5 ± 4.6	98.9 ± 2.5	98.9 ± 2.5
10	16.5 ± 4.6	95.3 ± 4.7	97.6 ± 3.4
25	16.5 ± 4.6	74.6 ± 2.6	70.6 ± 7.7
50	14.8 ± 5.3	54.1 ± 16.0	48.2 ± 20.2
Qwen 2.5-1.5B — Transition regime			
5	0.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
10	0.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
25	0.0 ± 0.0	94.6 ± 5.0	90.5 ± 7.1
50	0.0 ± 0.0	66.2 ± 14.4	54.2 ± 10.1
75	0.0 ± 0.0	38.3 ± 10.5	20.2 ± 12.6
100	0.0 ± 0.0	22.9 ± 14.5	6.7 ± 9.9

Table 4: Strict acceptance rate (%) in stable regime, Qwen 2.5-7B. Values are mean ± std across 5 seeds. Momentum achieves 0% across all K values.

K	Momentum	Linear	Quadratic
5	0.0 ± 0.0	66.7 ± 22.4	60.0 ± 41.8
10	0.0 ± 0.0	73.3 ± 27.5	60.0 ± 41.8
25	0.0 ± 0.0	73.3 ± 27.5	70.0 ± 44.7
50	0.0 ± 0.0	60.0 ± 34.6	80.0 ± 27.4
75	0.0 ± 0.0	90.0 ± 22.4	60.0 ± 54.8
100	0.0 ± 0.0	90.0 ± 22.4	80.0 ± 44.7

Linear prediction dominates. Linear prediction achieves 60–90% strict acceptance across all K values, substantially outperforming GPT-2 124M's stable-regime rates (5–24%). Remarkably, acceptance *does not degrade* with K: K=100 achieves 90% strict vs. K=5's 67%. This is likely because the 7B model near convergence has extremely flat loss landscapes—the linear predictor's extrapolation remains within the loss basin regardless of horizon. Adaptive acceptance is 100% across all K values and seeds.

Linear outperforms quadratic. Quadratic prediction (mean 60–80% strict) has substantially higher cross-seed variance (std 27–55%) than linear (std 22–35%). At small K, the extra degree of freedom in quadratic fitting sometimes overshoots; at large K, it sometimes helps. The linear predictor's simplicity and stability make it the preferred choice at 7B scale.

Seed 46 as informative outlier. Seed 46 achieves the lowest final validation loss (0.970) but the worst predictor acceptance at small K (50% linear, 40% quadratic at K=5). Investigation reveals this is *not* due to regime distribution—seed 46 has 13 stable checkpoints, exactly at the 5-seed mean. Rather, the loss curve exhibits non-monotone micro-oscillations in the stable regime: the model has converged so tightly that validation loss fluctuates stochastically rather than decreasing monotonically, causing strict-criterion rejections. This observation suggests that at 7B, the strict acceptance criterion becomes unnecessarily conservative near convergence, and adaptive acceptance (which achieves 100%) is the appropriate metric.

5.3 The Universal Momentum Catastrophe

Table 5 quantifies the momentum predictor's failure mode across all three scales.

Table 5: Momentum predictor failure across all three model scales. "Predicted loss" is the validation loss of the model with momentum-extrapolated weights; "Actual loss" is the current checkpoint's validation loss. Momentum catastrophe is universal—not scale-dependent.

	GPT-2 124M			**Qwen 1.5B**			**Qwen 7B**		
K	Pred.	Act.	Ratio	Pred.	Act.	Ratio	Pred.	Act.	Ratio
5	201	1.64	122×	142	0.82	173×	222	0.97	227×
10	360	1.64	219×	250	0.82	305×	347	0.97	357×
25	1284	1.64	782×	581	0.82	709×	681	0.97	700×
50	4505	1.64	2742×	1153	0.82	1407×	1259	0.97	1296×
75	9988	1.64	6077×	1819	0.82	2218×	1901	0.97	1957×
100	17691	1.64	10764×	2468	0.82	3009×	2601	0.97	2678×

The momentum catastrophe is universal across all three scales. At 124M, predicted losses exceed actuals by 122× at K=5, growing to 10,764× at K=100. The GPT-2 ratios are actually *higher* than both Qwen models at large K (10,764× vs. ∼3,000×): in absolute terms, GPT-2 momentum predictions reach 17,691 at K=100 vs. Qwen 7B's ∼2,601, likely reflecting GPT-2's longer exposure to stable/transition regimes where momentum accumulates larger update magnitudes. At 7B, the inflation ratios cluster more tightly across seeds (143−327× at K=5 across 5 seeds), consistent with more uniform parameter distributions in larger models. This is not a marginal degradation—it is a qualitative failure where the predicted weights lie far outside the region of the loss landscape explored during normal training.

By contrast, linear prediction at K=5 produces predicted losses differing from actuals by <0.001 at both scales. The 10^5-fold difference between momentum and linear predictions at the same depth demonstrates that the choice of extrapolation basis (optimizer state vs. observed trajectory) is the critical design decision, not model scale.

5.4 Cross-Seed Consistency

Table 6 reports the coefficient of variation (CoV) for acceptance rates across the five seeds.

Table 6: Coefficient of variation (%) for proximity acceptance rates across 5 seeds, Qwen 2.5-1.5B transition regime. Lower CoV indicates more consistent results across random seeds.

K	Linear CoV	Quadratic CoV
5	0.0%	0.0%
10	0.0%	0.0%
25	5.2%	7.9%
50	21.7%	18.6%
75	27.3%	62.5%
100	63.3%	149.1%

At short prediction horizons ($K \leq 10$), cross-seed results are perfectly consistent (CoV = 0%). Variance increases with K, as expected: longer predictions depend more on seed-specific trajectory details. The quadratic predictor shows higher variance than linear at large K, reflecting its sensitivity to the curvature of individual trajectories.

5.5 Cascade Evaluation

Cascaded predictions (Section 3.3) were evaluated from stable-regime checkpoints. At 1.5B scale, the scarcity of stable checkpoints (0–2 per seed) limits cascade evaluation. For seed 46, Pass 3 cascades encountered an out-of-memory condition. The limited cascade data available shows acceptance at short cascade depths (D=2, K=10) but rapid rejection at deeper cascades, consistent with error accumulation across prediction stages.

6 Discussion

6.1 The Universal Momentum Catastrophe

Our most significant finding is that momentum-based weight prediction—extrapolating Adam's exponential moving average of gradients—fails catastrophically at *every* model scale tested. At 124M parameters, momentum-predicted losses exceed actuals by 122× at K=5, growing to 10,764× at K=100. At 1.5B parameters, the pattern is similar: 173× at K=5 to 3,009× at K=100. At 7B parameters, the catastrophe continues: ∼227× at K=5 to ∼2,678× at K=100, with inflation tightening into a narrower band across seeds.

All three predictors operate in weight space, yet they differ in *how* they construct the extrapolation direction. The momentum predictor uses Adam's internal state ($m_t/\sqrt{v_t}$), which accumulates gradient information across the entire training history with exponential decay. The linear and quadratic predictors use finite differences of *actually observed* checkpoint parameters ($\theta_t - \theta_{t-\Delta}$), which implicitly capture the net effect of the learning rate schedule, gradient noise, and landscape curvature over the checkpoint interval.

The momentum predictor's universal failure reflects norm explosion: the extrapolated direction $K \cdot m_t/\sqrt{v_t}$ produces a displacement whose magnitude far exceeds the region of validity around the current point on the loss surface, regardless of model size. The finite-difference predictors are inherently bounded by the actual step sizes taken during training, providing a natural regularization that momentum lacks.

This finding has practical implications for weight nowcasting methods like WNN [Jang and Han, 2023] and NiNo [Knyazev et al., 2025], which learn to predict in weight space. Our results across three model scales (124M, 1.5B, 7B) demonstrate that optimizer-state extrapolation is fundamentally unsuitable for speculative weight prediction, and that methods must use trajectory-bounded extrapolation—extrapolating from observed weight deltas rather than accumulated gradient moments.

6.2 Non-Monotonic Regime Distribution

The regime distribution changes with model scale in a non-monotonic pattern. At 124M parameters, GPT-2 spends 34% of training in stable regime and only 4% in chaotic regime. At 1.5B parameters, Qwen spends 64% in chaotic regime and barely reaches stable (2.5% of checkpoints). At 7B parameters, the pattern reverses: Qwen 7B spends 48% in chaotic but recovers 34% stable—matching 124M.

This non-monotonic scaling has implications for the practical viability of speculative prediction. The 1.5B model represents a worst case: large enough to have complex optimization dynamics (prolonged chaotic phase) but not large enough for those dynamics to settle quickly. The 7B model, with its 4.7× more parameters, transitions through chaos faster in wall-clock terms and reaches a stable plateau where linear prediction achieves 60–90% strict acceptance—far above the 1.5B transition-regime rates (37%) and 124M stable-regime rates (24%).

Within comparable regimes, finite-difference predictors are monotonically *more accurate* at larger scales: linear prediction achieves 9% strict in GPT-2's transition, 37% in Qwen 1.5B's transition, and 60–90% in Qwen 7B's stable regime. Larger models have smoother loss landscapes, enabling more reliable extrapolation once the chaotic phase is passed.

6.3 Regime Detection as a Prerequisite for Prediction

The three-regime framework is essential, not merely convenient. In chaotic regimes, no predictor achieves meaningful acceptance rates. The regime detector prevents wasted computation by suppressing prediction attempts during these phases. More importantly, the transition from chaotic to stable regimes occurs at remarkably consistent training steps across seeds (±50 steps), suggesting that regime boundaries are properties of the optimization landscape rather than artifacts of random initialization.

6.4 Connection to ASC

The parallel between Leap+Verify and ASC [Waterland et al., 2014] runs deeper than analogy. In ASC, the recognizer identifies states from which prediction is "tractable and useful"—precisely the role of our regime detector. ASC's predictors learn from observed trajectory structure, as do our weight extrapolators. The trajectory cache stores verified state transitions; our acceptance criteria serve the same gatekeeping function. Even the scaling behavior has parallels: ASC found that prediction accuracy varies by program structure, just as Leap+Verify finds that prediction accuracy varies by model scale.

The key difference is verification cost. In ASC, verification requires executing the speculative segment and comparing end states—potentially as expensive as the original computation. In Leap+Verify, verification requires only a single forward pass on held-out data, which is $O(1)$ relative to the K gradient steps being predicted. This asymmetry makes Leap+Verify's verify-then-accept mechanism particularly favorable.

6.5 Limitations

Training was limited to 2000 steps on WikiText-103. The Qwen 1.5B model barely reached stable regime within this budget (0–2 stable checkpoints out of 40), limiting our ability to evaluate predictors in the regime where they are most useful. The 7B model largely resolves this concern by reliably reaching stable regime ($\sim$13 checkpoints), but longer training runs at 1.5B would clarify whether this scale eventually reaches stable as well.

The regime thresholds ($\tau_{\text{high}}, \tau_{\text{low}}$) were calibrated on GPT-2 124M and may require recalibration for larger models, where activation similarities tend to be higher overall. An adaptive thresholding scheme would improve generality.

Cascaded predictions (Pass 3) were infeasible at 7B due to memory constraints. We did not evaluate ensemble collapse (dynamic reduction of multi-seed training runs based on detected convergence), which is a planned extension of the framework.

7 Related Work

Weight prediction during training. The concept of predicting future weights to accelerate training dates to Kamarthi and Pittner [1999], who used Taylor series extrapolation. Sinha et al. [2017] trained neural predictors on weight histories. Recent work has produced increasingly sophisticated methods: WNN [Jang and Han, 2023] uses a learned nowcaster, NiNo [Knyazev et al., 2025] uses graph neural networks, and XGrad [Guan et al., 2024] constructs mathematical future-weight relationships for specific optimizers. PLP [Anonymous, 2024] applies linear prediction every 3 iterations. All these methods apply predictions unconditionally; Leap+Verify introduces regime-conditional prediction with verify-then-accept.

Training dynamics. Cohen et al. [2021] identified edge-of-stability via Hessian eigenvalues. Frankle et al. [2020] detected stable training via weight-space linear interpolation. Nanda et al. [2023] identified three mechanistic phases in grokking. Our activation-space cosine similarity provides a computationally cheap alternative that operates in real time.

Loss landscape geometry. Li et al. [2018] introduced filter-normalized visualization. Garipov et al. [2018] and Draxler et al. [2018] established mode connectivity. Keskar et al. [2017] linked batch size to basin sharpness. Foret et al. [2021] operationalized flat-minima seeking via SAM. These provide geometric context for why prediction difficulty varies by regime.

Adaptive optimization. The Lookahead optimizer [Zhang et al., 2019] maintains fast/slow weights with k-step lookahead, but always interpolates back (no acceptance criterion) and runs all k inner steps. Smith [2017] showed cyclical learning rates implicitly traverse different curvature regimes. RAdam [Liu et al., 2020] automatically transitions between SGD-like and Adam-like behavior based on variance estimates—a single-signal regime detection for one transition.

Speculative execution. Speculative decoding [Leviathan et al., 2023, Chen et al., 2023] and blockwise parallel decoding [Stern et al., 2018] provide the predict-then-verify template. ASC [Waterland et al., 2014, 2013] generalizes this to arbitrary sequential computation via trajectory-based execution with learned predictors.

Dynamical systems in training. Geiger et al. [2022] showed chaos is intrinsic to SGD. Tajanthan et al. [2022] connected Lyapunov exponents to Hessian eigenvalues. Saxe et al. [2014] derived exact training dynamics for deep linear networks, revealing phase-transition-like behavior. The Ensemble Kalman Filter literature [Evensen, 1994, Gao et al., 2011] provides analogies for regime-dependent ensemble behavior, where error growth concentrates in unstable regions.

8 Conclusion

We have introduced Leap+Verify, a framework for speculative weight prediction during neural network training that incorporates regime detection and verify-then-accept validation. Our experiments across three model scales (124M, 1.5B, 7B) reveal three key findings: (1) momentum-based weight prediction (optimizer-state extrapolation) fails catastrophically at *all* scales (100−10,000× loss inflation); (2) the distribution of training regimes varies non-monotonically with scale—1.5B spends 64% of training in chaotic regime while 7B recovers to 34% stable, matching 124M; and (3) finite-difference extrapolation from observed weight trajectories succeeds with *increasing* accuracy at larger scales, achieving 60–90% strict acceptance at 7B vs. 9–37% at smaller scales. Together, these findings suggest that weight-space prediction methods must use trajectory-bounded extrapolation rather than optimizer-state extrapolation, and that the regime availability bottleneck observed at 1.5B is not universal—at 7B, the bottleneck lifts and linear prediction becomes highly effective.

The regime detection framework—classifying training into chaotic, transition, and stable phases using activation cosine similarity—provides a computationally cheap signal that generalizes the edge-of-stability and linear mode connectivity observations in prior work. The verify-then-accept mechanism, transplanted from speculative decoding and ASC, ensures that bad predictions have zero cost: rejected leaps leave the training trajectory unmodified.

Future work will extend the framework in three directions: (1) adaptive regime thresholds that recalibrate with model scale, (2) ensemble collapse—dynamically reducing multi-seed training runs when regime detection indicates cross-seed convergence, and (3) evaluation at larger scales to determine whether the non-monotonic regime pattern continues or stabilizes.

Acknowledgments

The author thanks Amos Waterland for creating the ASC architecture and for formative discussions on trajectory-based speculative computation that seeded this research direction over a decade ago. The author contributed to the original ASC work at Harvard and is acknowledged in Waterland et al. [2014]. Brian Cremeans is acknowledged for discussions on eigenvalue sign toggling in Lorenz system dynamics and its analogy to training regime transitions. Claude (Anthropic) provided extensive assistance with experimental implementation, GPU infrastructure management, data analysis, literature survey, and manuscript preparation.

This work used GPU compute from vast.ai. Experiments were conducted on NVIDIA A100 40GB (124M, 1.5B) and A100 80GB (7B) GPUs.

Code availability. All code, experiment scripts, and paper source are available at `https://github.com/jmcentire/leap-verify` (DOI: 10.5281/zenodo.18739387).

References

Anonymous. Enhancing deep neural network training efficiency and performance through linear prediction. *Scientific Reports*, 14:15443, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations (ICLR)*, 2021.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A Hamprecht. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning (ICML)*, 2018.

Geir Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99(C5):10143–10162, 1994.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations (ICLR)*, 2021.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, 2020.

Chao Gao, Han Wang, Ensheng Weng, S Lakshmivarahan, Yanfen Zhang, and Yiqi Luo. Assimilation of multiple data sets with the ensemble Kalman filter to improve forecasts of forest carbon dynamics. *Ecological Applications*, 21(5):1461–1473, 2011.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of DNNs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Mario Geiger, Leonardo Petrini, and Matthieu Wyart. Chaotic dynamics are intrinsic to neural network training with SGD. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Lei Guan, Dongsheng Chen, et al. XGrad: Boosting gradient-based optimizers with weight prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Jinhyeok Jang and Woo-hun Han. Learning to boost training by periodic nowcasting near future weights. In *International Conference on Machine Learning (ICML)*, 2023.

Sagar V Kamarthi and Stefan Pittner. Accelerating neural network training using weight extrapolations. *Neural Networks*, 12(9):1285–1299, 1999.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations (ICLR)*, 2017.

Boris Knyazev, Maude Drozdzal, Graham W Taylor, and Adriana Romero. Accelerating training with neuron interaction and nowcasting networks. In *International Conference on Learning Representations (ICLR)*, 2025.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, 2023.

Aitor Lewkowycz, Yasaman Bahri, Ethan Dyer, Jascha Sohl-Dickstein, and Guy Gur-Ari. The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*, 2020.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations (ICLR)*, 2020.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations (ICLR)*, 2023.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Abhishek Sinha, Mausoom Sarkar, Avisek Mukherjee, and Balaji Krishnamurthy. Introspection: Accelerating neural network training by learning weight evolution. In *ICLR Workshop*, 2017.

Leslie N Smith. Cyclical learning rates for training neural networks. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Vinoj Tajanthan et al. A chaos theory approach to understand neural network optimization. *arXiv preprint*, 2022.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

Amos Waterland, Elaine Angelino, Ekin D Cubuk, Efthimios Kaxiras, Ryan P Adams, Jonathan Appavoo, and Margo Seltzer. Computational caches. In *Proceedings of the 6th International Systems and Storage Conference (SYSTOR)*. ACM, 2013. doi: 10.1145/2485732.2485750.

Amos Waterland, Elaine Angelino, Ryan P Adams, Jonathan Appavoo, and Margo Seltzer. ASC: Automatically scalable computation. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pages 575–590. ACM, 2014. doi: 10.1145/2541940.2541985.

Michael R Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.